Московский институт электроники и математики имени А. Н. Тихонова

Кафедра информационной безопасности киберфизических систем

Москва 2025

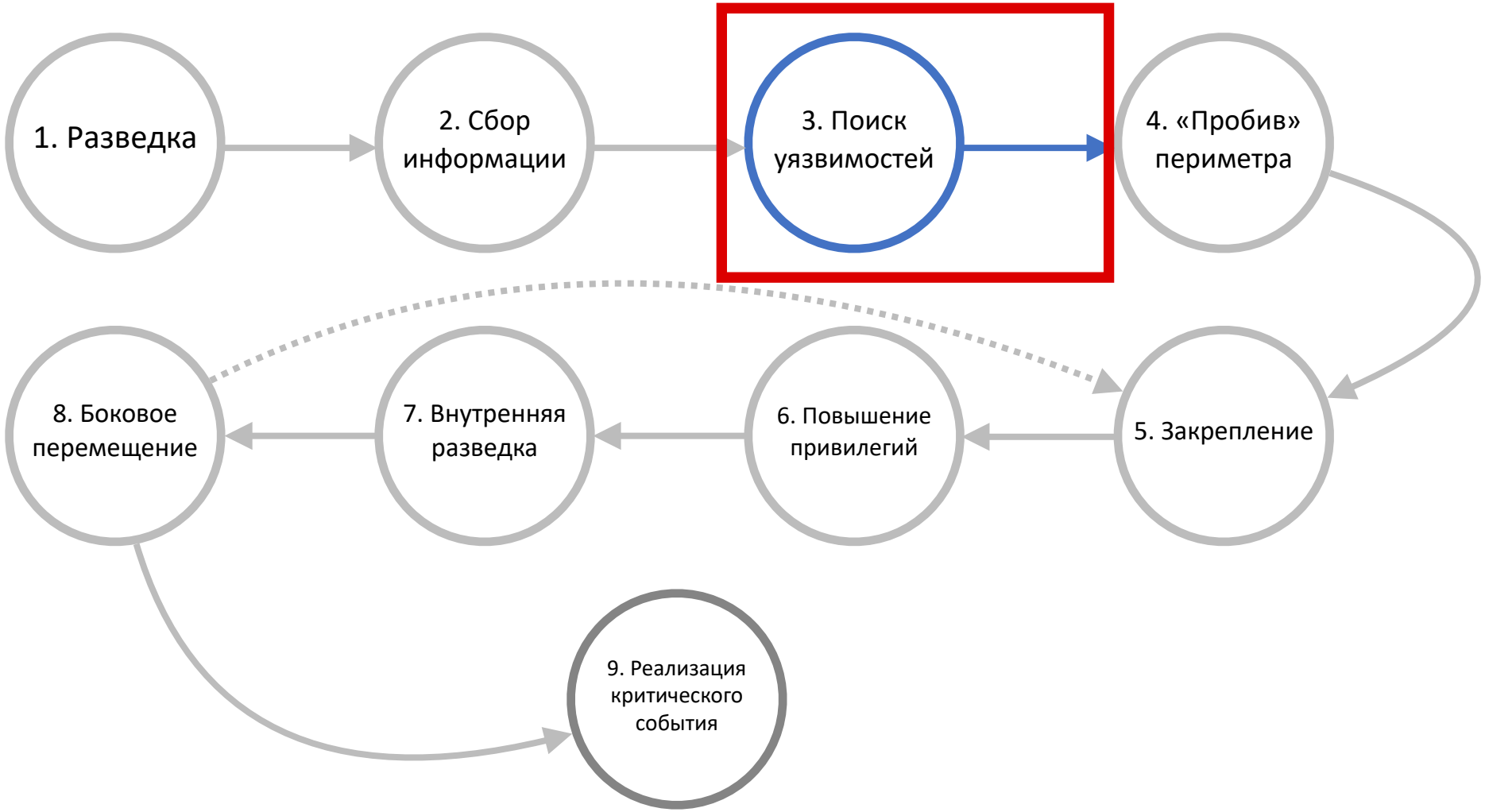
Лекция 5: Серверные уязвимости веб-приложений Часть 2: Аутентификация и авторизация

Курс: Технологии пентестинга

Автор: Космачев Алексей Алексеевич

План лекции

1. Основные понятия
2. Атаки на аутентификацию
3. Сессии
4. JSON Web Tokens (JWT)

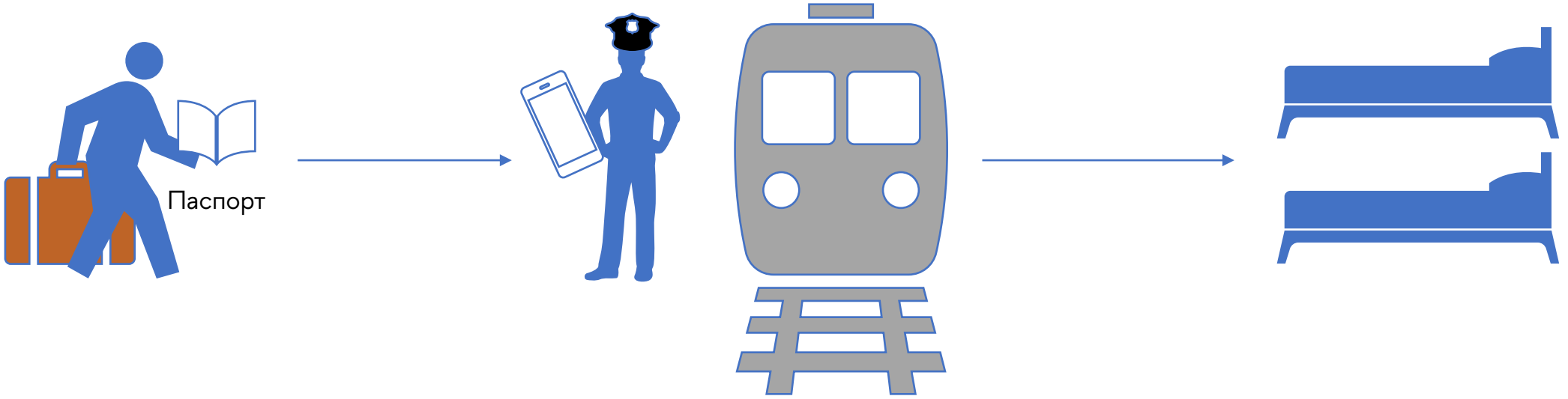

1. Разведка
2. Сбор информации
3. Поиск уязвимостей
4. «Пробив» периметра
5. Закрепление
6. Повышение привилегий
7. Внутренняя разведка
8. Боковое перемещение
9. Реализация критического события

Основные понятия

Идентификация, аутентификация, авторизация

- **Идентификация** — процедура, в результате выполнения которой для субъекта идентификации выявляется его идентификатор, однозначно определяющий этого субъекта в информационной системе.
- **Аутентификация** — процедура проверки подлинности, например проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.
- **Авторизация** — предоставление определенному лицу или группе лиц прав на выполнение определенных действий.

Идентификация, аутентификация, авторизация

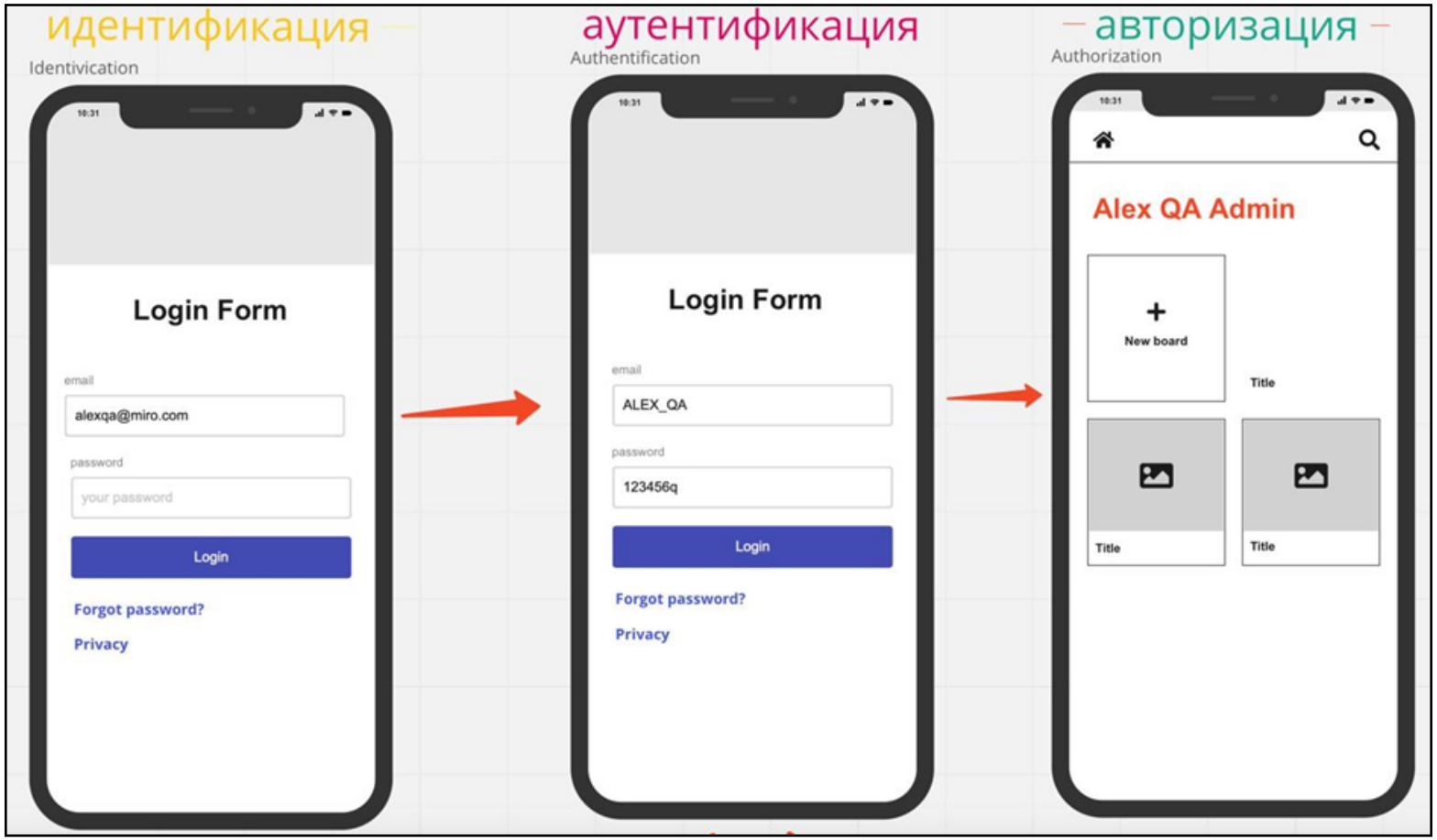


Идентификация
(Предъявление паспорта)

Аутентификация
(Проверка наличия билета)

Авторизация
(Возможность занять свое место)

Идентификация, аутентификация, авторизация



https://habr.com/ru/articles/720842/

Сессии и JWT

Критерий	Сессии	JSON Web Tokens (JWT)
Место хранения состояния	Серверная память или хранилище (БД)	Клиентская сторона (localStorage, cookies)
Масштабируемость	Требует синхронизации сессий между серверами	Stateless — не требует хранения состояния на сервере
Производительность	Запросы к хранилищу для проверки сессии	Быстрая проверка подписи токена без запросов к БД
Межсерверное взаимодействие	Сложно использовать в микросервисной архитектуре	Идеально для микросервисов (самодостаточный токен)
Отзыв	Легко отозвать через сервер (удаление сессии)	Сложнее отозвать (требуются черные списки или короткие TTL)

Атаки на аутентификацию

Аутентификация и факторы

- **Аутентификация** — это процесс подтверждения личности пользователя или клиента. Веб-сайты потенциально уязвимы для любого, кто подключен к интернету. Поэтому надежные механизмы аутентификации являются неотъемлемой частью эффективной веб-безопасности.

Аутентификация имеет разные количества (двух-трех-факторная аутентификация и т.д.) и типы факторов:

1. Знание (Knowledge) - что-то, что ты **знаешь** (пароль, ответ на секретный вопрос)
2. Владение (Possession) - что-то, чем ты **владеешь** (телефон, физический токен)
3. Свойство (Inherence) - что-то, что **является частью тебя** (биометрия, характер поведения)

Возможное влияние

- Захват аккаунта (Account Takeover, ATO)
- Кража персональных данных
- Повышение привилегий в приложении
- Повышение шансов на RCE

Password-based login: методы атаки

1. Использование слабых паролей и паролей по умолчанию
   admin:admin
   postgres:postgres
   root:root
   tomcat:tomcat

Top 20 most common passwords according to NordPass

Rank	Password	Count of password uses
1	123456	3,018,050
2	123456789	1,625,135
3	12345678	884,740
4	password	692,151
5	qwerty123	642,638

Default Credentials for Apache Tomcat

default-creds-tomcat.txt

```
admin : admin
ADMIN : ADMIN
admin : j5Brn9
admin : None
admin : tomcat
cxsdk : kdsxc
j2deployer : j2deployer
ovwebusr : OvW*busr1
QCC : QLogic66
role : changethis
role1 : role1
role1 : tomcat
root : root
tomcat : changethis
tomcat : s3cret
tomcat : tomcat
xampp : xampp
```

Защита:

- Проверка сложности пароля
- Смена паролей по умолчанию

Password-based login: методы атаки

2. Credential Stuffing - переиспользование пар логин-пароль, "слитых" в открытых/закрытых источниках

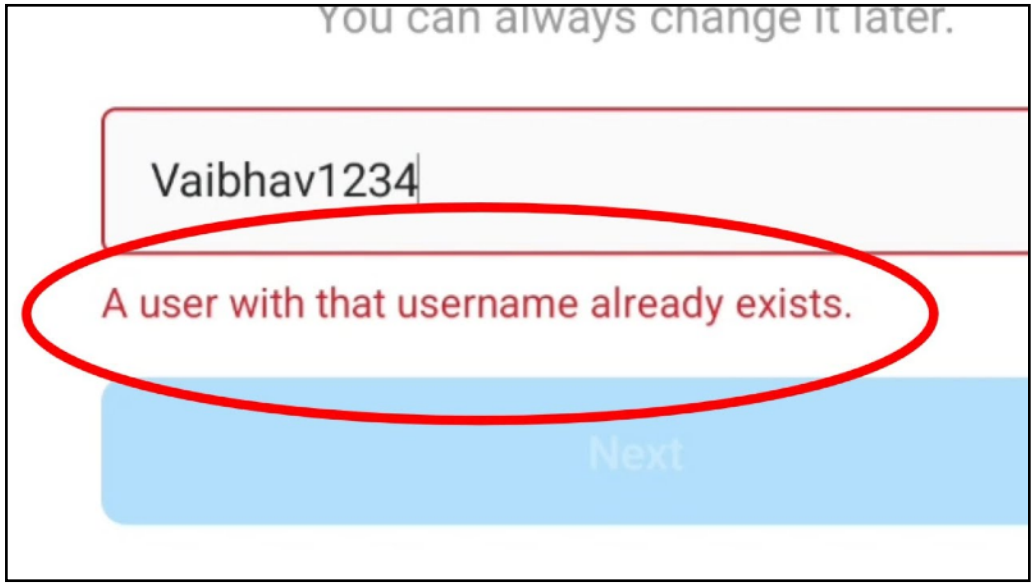
Защита:

- Регулярная смена паролей

Password-based login: методы атаки

3. Username Enumeration - возможность определить существование пользователя в системе исходя из:

- Различных текстов ответа приложения
- Различных кодов ответа
- Незначительных опечаток в «одинаковых» ответах приложения
- Разного времени ответа приложения



Защита:

- Возвращение одинакового ответа вне зависимости от существования пользователя в системе

Password-based login: методы атаки

4. Перебор логинов/паролей (Bruteforce)

Метод защиты	Вариант обхода
Блокировка по IP	• Сброс счетчика путем входа в валидный аккаунт • Маскировка настоящего IP-адреса (напр. X-Forwarded-For) • Использование распределенной системы перебора
Блокировка учетной записи	Создает возможность User Enumeration Делает возможным множественную блокировку УЗ
CAPTCHA	Эксплуатация мисконфигураций CAPTCHA (модификация, переиспользование, игнорирование и пр.)

Password-based login: методы атаки

5. «Умный» перебор - эксплуатация особенностей языков программирования и технологий, используемых в приложении

Примеры:

- В некоторых случаях мы можем отправить не один пароль, а массив паролей, каждый из которых будет проверяться приложением
- В GraphQL существуют алиасы, с помощью которых можно перебрать несколько значений одним запросом

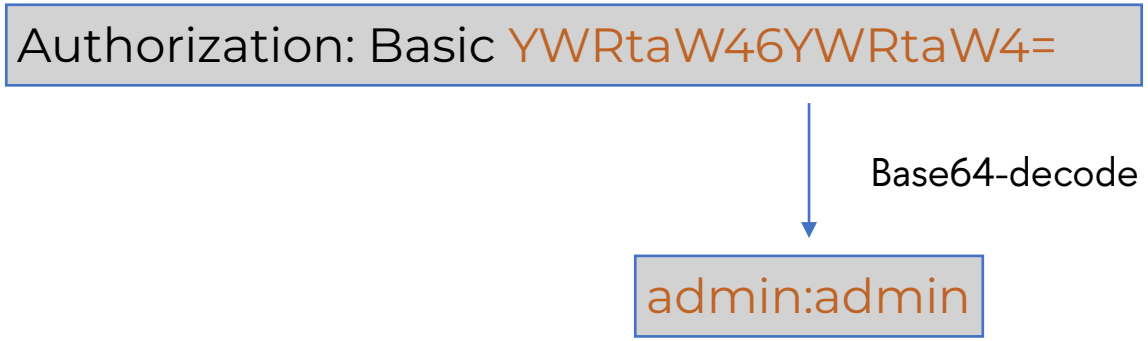
```
{"login":"admin","password":["1234","admin"]}
```

```
{
  empireHero: hero(episode: EMPIRE) {
    name
  }
  jediHero: hero(episode: JEDI) {
    name
  }
}
```

```
{
  "data": {
    "empireHero": {
      "name": "Luke Skywalker"
    },
    "jediHero": {
      "name": "R2-D2"
    }
  }
}
```

Password-based login: методы атаки

6. Перебор в базовой аутентификации (Basic Authentication)



Защита:

- Отказаться от базовой аутентификации

Client-Side Authentication

(Так делать не нужно никогда)

Мультифакторная авторизация: методы атаки

1. Слабый генератор OTP (One Time Password)
(напр. Стандартный Random в JAVA)

Слабый генератор:

```
SEED = 123123
rand(SEED) = one_string
rand(SEED) = one_string
```

Более надежный генератор:

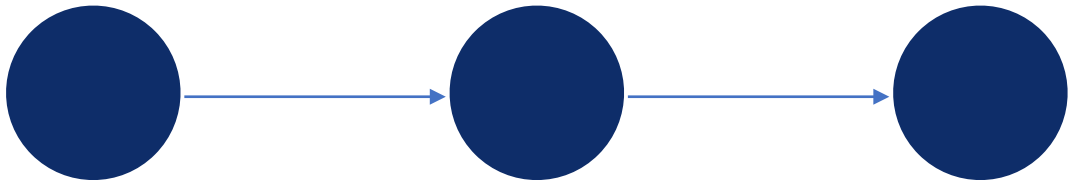
```
SEED = 123123
rand(SEED) = one_string
rand(SEED) = another_string
```

Защита:

- Использование надежных приложений (напр. Google Authenticator) и функций для генерации OTP

Мультифакторная авторизация: методы атаки

2. Некорректное состояние сессии



Защита:

- Корректный контроль состояния сессии

Мультифакторная авторизация: методы атаки

3. Account Hijack

https://example.com/2nd-stage?id=2

Защита:

- Привязка OTP к идентификатору
- Изменение логики

Мультифакторная авторизация: методы атаки

4. Перебор OTP (Bruteforce)

Метод защиты	Вариант обхода
Инвалидация временной сессии	Автоматизация прохождения первого фактора
Инвалидация OTP после N неверных попыток	Запрос нового кода и перебор N первых кодов (вероятностный подход)
Прочие техники защиты и обхода из атак на аутентификацию	

Мультифакторная авторизация: методы атаки

5. Flood

Защита:

- Лимит на отправку OTP

Прочие механизмы аутентификации: Кастомная аутентифиакационная cookie

1. Cookie guessing - подмена/угадывание/подбор
2. Insecure deserialization

Прочие механизмы аутентификации: Восстановление пароля

1. Token guessing - слабая энтропия или логические ошибки
2. SQLi Ultimate Power - легкий ATO через SQLi (почему?)
3. Host Header attacks
4. Exotic exploits (e.g. dangling markup injection)
5. User Enum
6. Flood

Прочие механизмы аутентификации: Смена пароля

1. Broken Access Control (смена пароля другому пользователю)
2. Logic Flaws (перебор пароля без блокировки УЗ)
3. Prototype Pollution (Server-Side)
4. Mass Assignment

Прочие механизмы аутентификации: Кастомная CAPTCHA

1. CAPTHCA guessing (угадывание, получение ответа от приложения)
2. CAPTCHA modify (замена значения на необходимое)
3. CAPTCHA reuse (переиспользование CAPTCHA)
4. CAPTCHA DoS (возможность испортить единичную CAPTCHA или группу по идентификатор, делая невозможным ее правильный ввод)

Прочие механизмы аутентификации: Регистрация

1. Mass Assignment
2. Prototype Pollution (Server-Side)

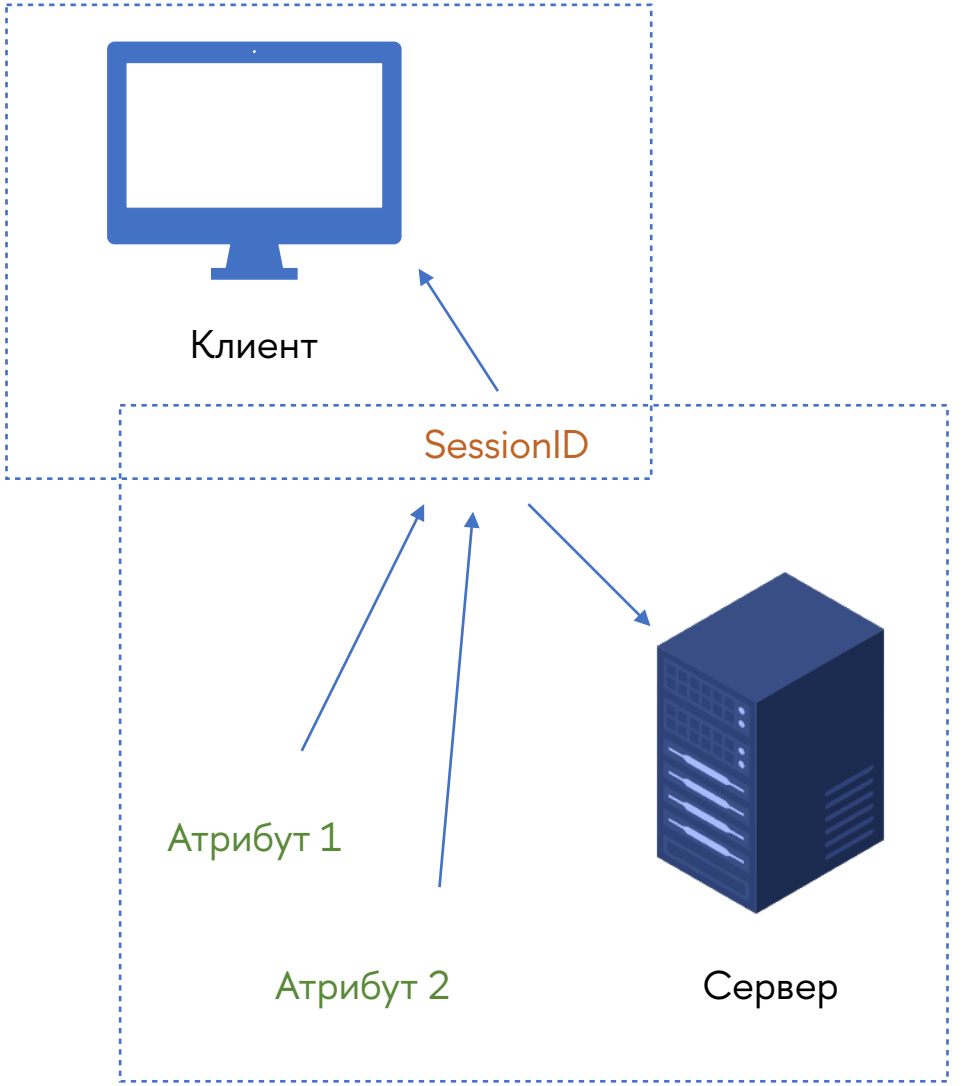
Сессии

Определение

- **Сессия (Session)** — это механизм, который позволяет серверу сохранять информацию о взаимодействии с конкретным пользователем в течение определенного периода времени (пока пользователь активен на сайте).

Проще говоря, это способ "помнить" пользователя между отдельными HTTP-запросами.

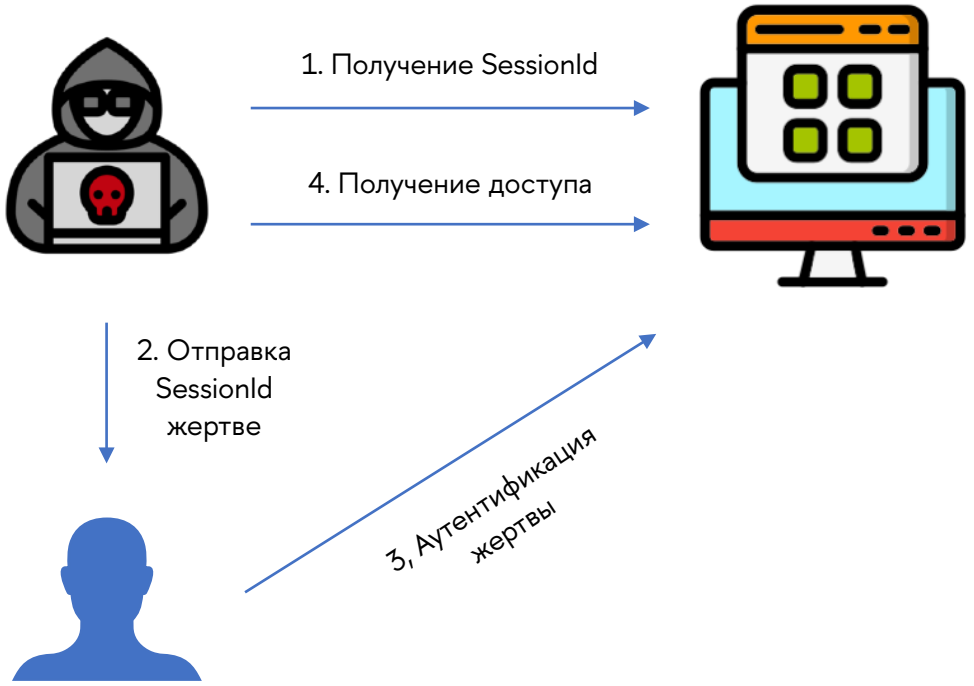
На стороне сервера, идентификатору сессии присваиваются атрибуты (свойства, состояния)



Session Fixation

Фиксация Сессии (Session Fixing) позволяет злоумышленнику перехватить действующий сеанс пользователя. Причина уязвимости в способе управления идентификатором сеанса веб-приложения.

- Атакующий может получить идентификатор сессии и отправить жертве. После того, как жертва аутентифицируется в системе, атакующий получит валидную сессию жертвы



Атака реализуется, если выполняются оба условия:

1. Приложение не выдает нового идентификатора сессии после успешной аутентификации пользователя
2. Атакующий может выставить жертве идентификатор сессии (через client-side уязвимость или иными способами)

Защита:

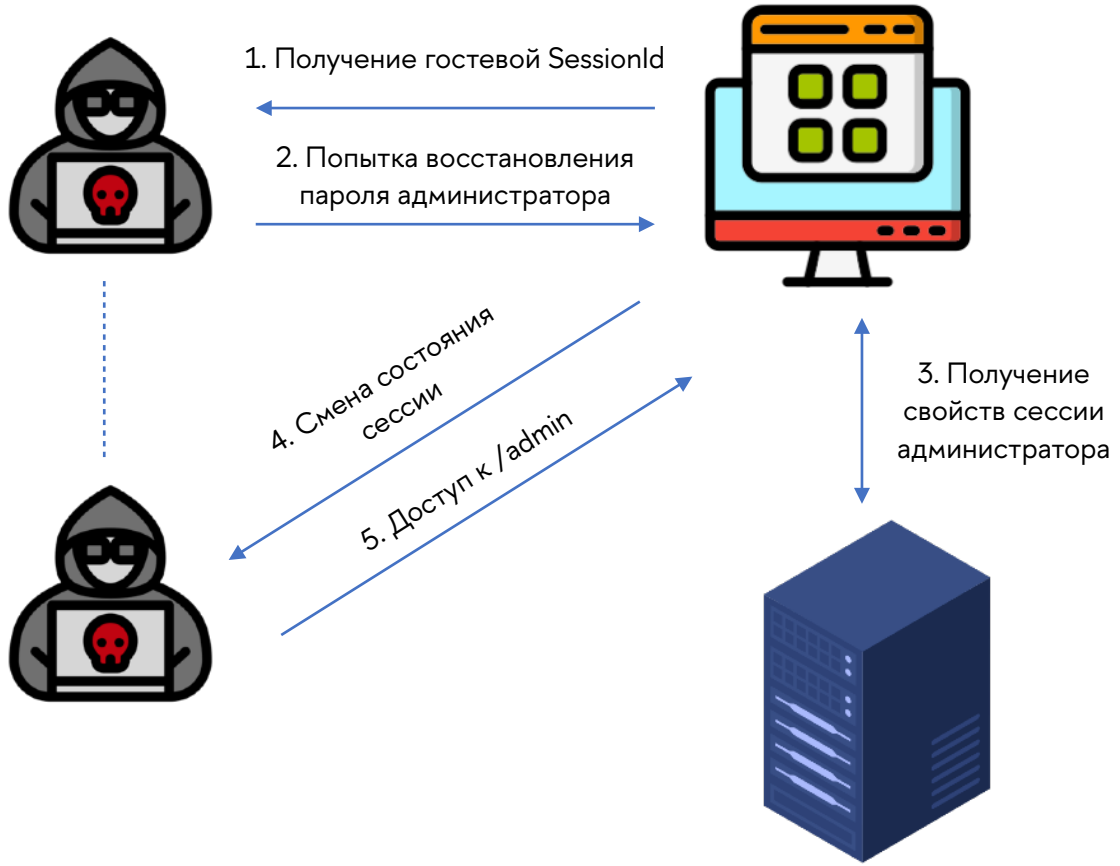
- Выдача нового SessionID после аутентификации

Session Puzzling

Session Puzzling (Session Variable Overloading) — уязвимость, которая возникает, когда переменная сеанса приложения используется для более одного назначения.

- В результате эксплуатации атакующий получает доступ к свойствам сессии другого пользователя

Защита: использовать переменную сеанса только для одного назначения



Improper Session Invalidation

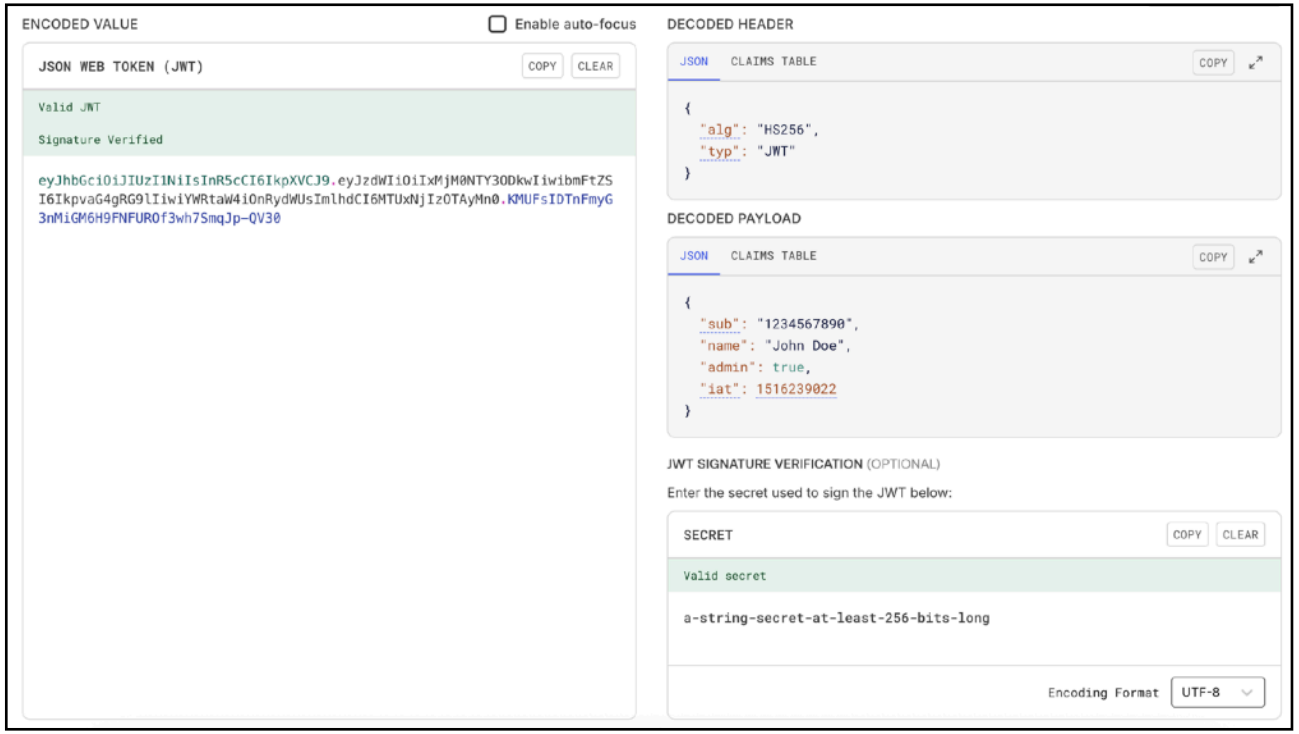
- После выхода из аккаунта сессия остается валидной
- Если сессия будет украдена, то нет способа сделать ее невалидной кроме прямого вмешательства

Защита: корректная инвалидация сессии после выхода из аккаунта или по истечении времени

JSON Web Tokens (JWT)

Определение

- **JSON Web Tokens (JWT)** - стандартизированный формат для передачи криптографически подписанных данных JSON между системами. Теоретически они могут содержать любые данные, но чаще всего используются для передачи информации («claims») о пользователях в рамках механизмов аутентификации, управления сеансами и контроля доступа.



https://www.jwt.io/

Структура

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWUsImlhdCI6MTUxNjIzOTAyMn0.KMUFsIDTnFmyG3nMiGM6H9FNFUROf3wh7SmqJp-QV30

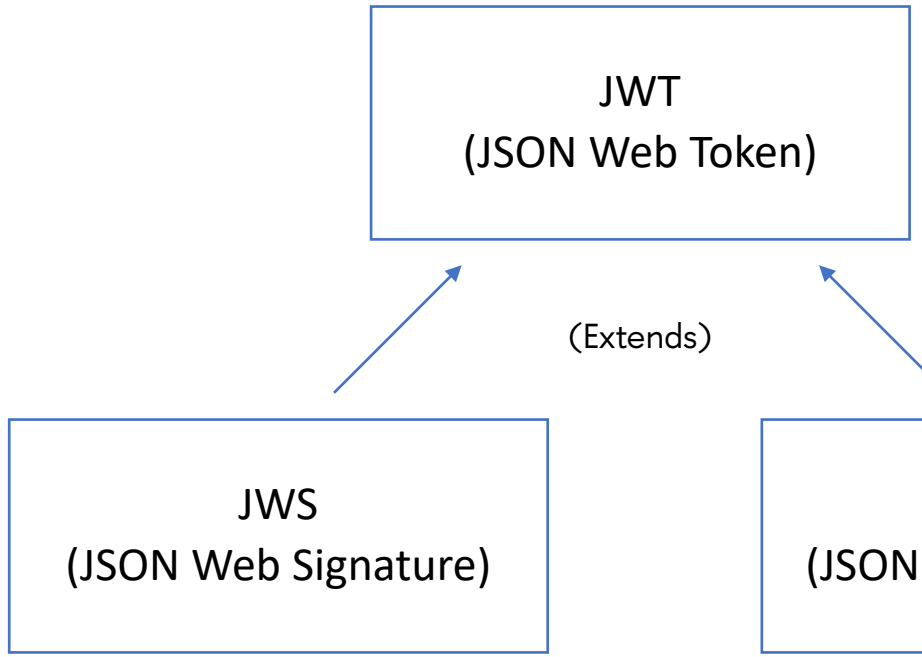
Base64-decode

```
{
  "alg": "HS256",
  "typ": "JWT"
}

{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239022
}

<signature>
```

JWT vs JWS vs JWE



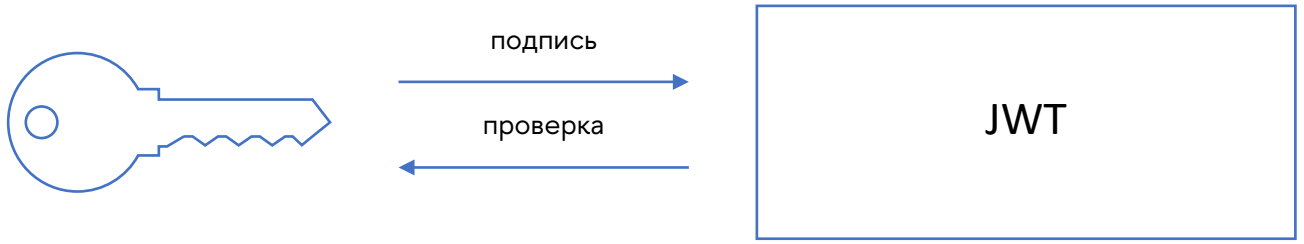
- Базовый формат передачи данных

- Стандарт подписи JWT при помощи секретного(-ых) ключа(-ей)
- **Обычно имеют в виду именно это, когда говорят о JWT**

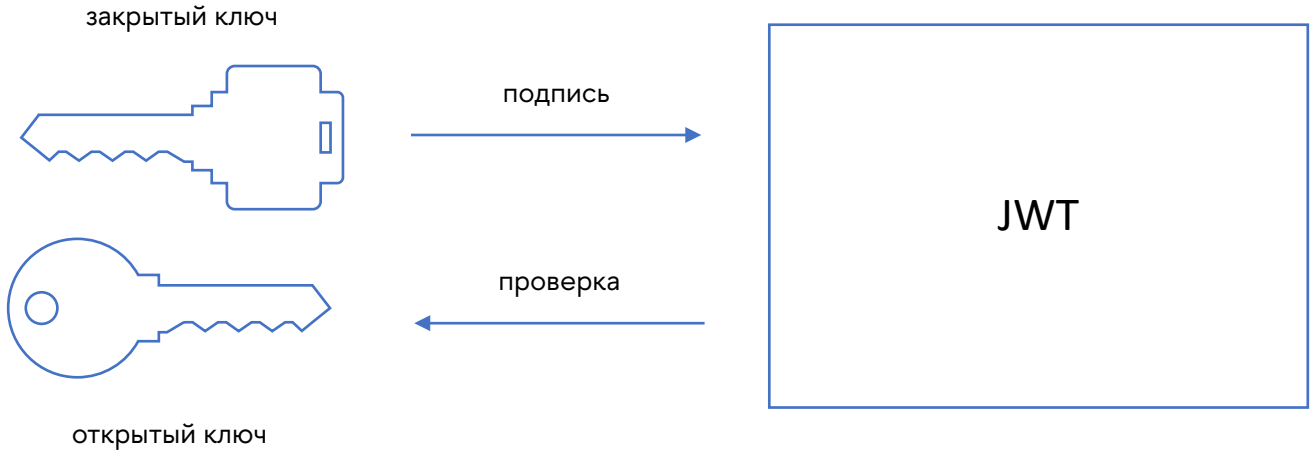
- Стандарт шифрования Payload-части
- Обычно имеет формат header.encrypted_key.iv.ciphertext.tag

Симметричные и асимметричные алгоритмы подписи

- Симметричный (HS256 - HMAC+SHA256) - один ключ используется и для подписи, и для проверки подписи

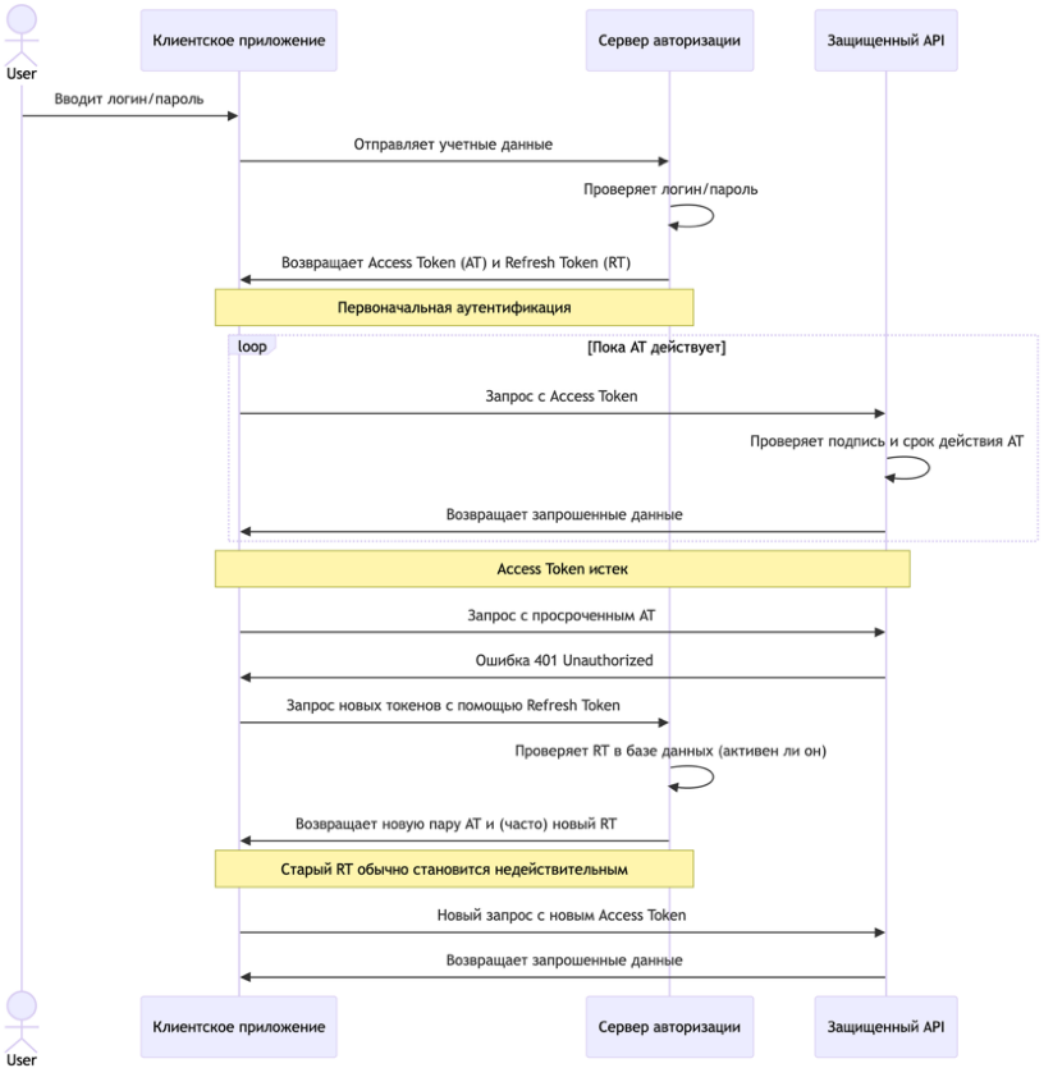


- Асимметричный (RS256 - RSA+SHA256) - закрытый ключ используется для подписи, открытый - для проверки подписи



Access и Refresh-токены

- Способ управления «долгосрочной» авторизацией пользователя
- В UI обычно будет применен после нажатия кнопки «запомнить меня»
- Идея: при аутентификации выдается два токена: Access и Refresh
- Токены помещаются в разные места и имеют разный срок жизни
- Access Token имеет короткий срок жизни и используется для авторизации
- Refresh Token имеет долгий срок жизни и используется для получения новой пары Access и Refresh-токенов
- Утечка Refresh-токена несет серьезный ущерб



Атаки на JWT: влияние

- Сильный контроль над приложением
- Обход аутентификации
- Повышение привилегий внутри приложения
- Account Takeover (ATO)
- Представление другим пользователем
- Вектор для других атак

Атаки на JWT: отсутствие проверки подписи

- Вместо метода verify() используется метод decode()
- Подпись не проверяется
- Можно записывать любые данные

```
token.decode()
```

```
token.verify()
```

Защита: проверять подпись JWT

Атаки на JWT: поддержка алгоритма none

- Приложение полностью доверяет токену от клиента (принцип JWT)
- Приложение не ограничивает используемые алгоритмы
- Происходит замена на алгоритм none и подпись «отрезается», оставляя при этом завершающую точку

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```



```
{
  "alg": "none",
  "typ": "JWT"
}
```

Структура JWT: header.payload.

Защита: отключить поддержку алгоритма none

Атаки на JWT: перебор ключей HS256

- Алгоритм HS256 - симметричный
- Если используется слабый ключ, его можно эффективно подобрать

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
hashcat -a 0 -m 16500 <jwt> <wordlist>
```

```
hashcat -a 0 -m 16500 <jwt> <wordlist> --show
```

Защита: использовать надежные ключи

https://github.com/wallarm/jwt-secrets/blob/master/jwt.secrets.list

JWK, JKU, KID

- **jwk (JSON Web Key)** - формат представления ключа в JSON-формате
- **jku (JSON Web Key Set URL)** - ссылка, по которой можно получить множество ключей и выбрать из них необходимый
- **kid (Key ID)** - идентификатор, позволяющий однозначно определить ключ из множества. В некоторых случаях позволяет указать путь до файла с ключом

Атаки на JWT: использование самоподписанного ключа (jwk)

1. Приложение использует асимметричный алгоритм (RS256)
2. Атакующий генерирует пару ключей
3. Атакующий подписывает JWT собственным приватным ключом, а публичный помещает в токен в jwk
4. Приложение использует ключ из jwk для проверки подписи => доверяет токену

```
{
   "kid": "ed2Nf8sb-sD6ng0-scs5390g-fFD8sfxG",
   "typ": "JWT",
   "alg": "RS256",
   "jwk": {
      "kty": "RSA",
      "e": "AQAB",
      "kid": "ed2Nf8sb-sD6ng0-scs5390g-fFD8sfxG",
      "n":
"yy1wpYmffgXBxhAUJzHHocCuJolwDqql75ZWuCQ_cb33K2vh9m"
   }
}
```

Защита: использовать белый список публичных ключей для проверки

Атаки на JWT: использование самоподписанного ключа (jku)

1. Приложение использует асимметричный алгоритм (RS256)
2. Атакующий генерирует пару ключей
3. Атакующий подписывает JWT собственным приватным ключом, а публичный помещает в JWK Set на своем сервере
4. В параметре jku указывается ссылка на JWK Set, а также изменяется kid
5. Приложение использует ключ из JWK Set, расположенный по URL из jku для проверки подписи => доверяет токену

Защита: принимать ключи только из доверенных источников

```
{
   "keys": [
      {
         "kty": "RSA",
         "e": "AQAB",
         "kid": "75d0ef47-af89-47a9-9061-7c02a610d5ab",
         "n": "<some_val>"
      },
      {
         ...
      }
   ]
}
```

```
{
   "jku":"https://exploit.domain/keys.txt",
   "kid":"75d0ef47-af89-47a9-9061-7c02a610d5ab",...
}
```

Атаки на JWT: использование самоподписанного ключа (kid)

1. Приложение использует симметричный алгоритм (HS256)
2. В значение kid можно указать путь до файла
3. Атакующий может подписать токен любым файлом, если знает его содержимое
4. Можно использовать /dev/null и подписать пустым ключом
5. Также, возможна эксплуатация SQLi через данный параметр

```
{
   "kid":"/dev/null",
   ...
}
```

Защита: не позволять принимать произвольный путь; защититься от SQLi

Атаки на JWT: algorithm confusion

Идея: смена ассиметричного алгоритма на симметричный. В качестве подписи использовать публичный ключ.

Этапы эксплуатации:

1. Получение публичного ключа:
   - в публично доступных директориях
   - вычислить на основе выданных токенов
2. Перевод в необходимый формат (JWK, X.509 PEM, и т.д.) - ключ должен побайтово совпадать с имеющимся на сервере
3. Создание JWT с alg HS256, подпись токена полученным ключом

/jwks.json
/.well-known/jwks.json
...

docker run --rm -it portswigger/sig2n <token1> <token2>
https://github.com/silentsignal/rsa_sign2n (jwt_forgery.py)

Защита:

- Проверка соответствия используемому алгоритму

JWT Invalidation

Так как JWT - Stateless (не имеет состояния на стороне сервера) существуют проблемы в его инвалидации.

Некоторые способы:

- Указание очень короткого времени жизни
- Черный список невалидных токенов
- Механизмы инвалидации токенов по идентификатору (нужны триггеры, например, повторное использование refresh-токена)
- Смена ключа подписи (радикальный метод)

Защита:

- Проверка соответствия используемому алгоритму

Дополнительные материалы для изучения

1. Oauth & OpenID Connect (+методы атак)
2. Инъекция сертификатов X.509 в JWT (CVE-2017-2800, CVE-2018-2633)

@LEXA_MALOSPAAL

@HUN7_0R_B3_HUN73D


ВЫСШАЯ • ШКОЛА • ЭКОНОМИКИ •